DynaDoodle

An example project to accompany the *NXApp* article
Branching Out With Dynamic Loading

Andrew Vyrros
Codeworks
av@codeworks.com

Overview
This project is meant to give you a concrete introduction to the concepts of dynamic loading. It implements a simple graphical display app. The app presents a selection of silly doodle drawings. These doodles are dynamically loaded at launch time from external modules. The default modules are contained in the app package. Additional doodles can be placed in a directory named **DynaDoodle** inside **~/Library**, **/LocalLibrary**, or **/NextLibrary**.

DynaDoodle shows you how to set up a project for dynamic loading. It demonstrates the way to create an abstract superclass for loaded classes, and the technique of distributing subclasses among individual bundle projects. It also explains how to search for module packages, and how to defer loading and instantiating until needed.

Technical details
The central object is an instance of the class *Manager*. It acts as NXApp's delegate, and handles basic application control duties. At launch time, it searches for doodle modules. For each doodle module it finds, it creates an instance of the class DoodleBundle.

DoodleBundle is a subclass of NXBundle geared specifically for managing doodle modules. Each DoodleBundle is responsible for a one module with a single loadable code file. DoodleBundle loads the class definitions in its code file into the Objective C class hierarchy. It defers this until requested, so that modules are not loaded until needed. Once loaded, DoodleBundle creates an instance of the module's principal class, the Doodle itself.

A *Doodle* is just a subclass of View for drawing silly images. It has foreground and background colors, and its own custom control View (sort of like a mini-inspector panel). Every module contains a single subclass of Doodle to do the actual drawing. The example supplies several different variations. Each of these has its own bundle project as a subproject of the main project. When the app is made, these are built as individual module packages and placed in the main app package.

The bundle projects are Checkerboard, Face, Smile, Spiral, and Star. They aren't meant to demonstrate any concepts, they just provide the Doodle subclasses for Manager to load. Most of the content is just silly non-optimized drawing code.

Points of interest

· Searching for modules: **Manager.m**, **-createBundlesAndLoadModules:** and **-createBundlesForDirectory:loadModules:**.

· Loading modules and instantiating Doodles: **Manager.m**, **-takeCurrentDoodleFrom:**; **DoodleBundle.m**, **-doodle**.

· Deferred loading of modules: **Manager.m**, **-appDidInit:** and **-createBundlesForDirectory:loadModules:**; **DoodleBundle.m**, **-doodle**.

· Getting localized names and descriptions of dynamic modules: **DoodleBundle.m**, **-doodleName** and **-doodleDescription**.

· Using custom nibs with dynamic modules: **Doodle.m**, **-customControlView**, **-loadNib** and **-didLoadNib**.

· Putting additional classes in a module: **Face.bproj**, **Face.m, EyePair.m**, and **Mouth.m**.

Suggested modifications

· Change Manager so that, rather than deferring the loading of modules until they are needed, it loads all modules at launch time.

· Modify Manager so that it lets the user choose the default doodle that first appears at launch time.

· Add your own new Doodles. To create a new Doodle module, first add a new bundle project in Project Builder. The project should contain at least one class, a custom subclass of Doodle. This subclass needs to override the **drawSelf::** method to perform its special drawing. It will also probably override **initFrame:** and **didLoadNib**, and may add some action methods for custom controls.

The bundle project should have a nib file named ***ClassName*.nib**, where *ClassName* is the name of the Doodle subclass. The nib will have the custom Doodle as owner, with at least the customControlView outlet connected to a Box or similar view. The project will also need a **Doodle.strings** file, which contains the strings for the Doodle name and description. You can generate this with **genstrings** and #define statements similar to those at the beginning of the other custom Doodle sources, or just copy one of the files and substitute your own values.